Emerald Insight

Program

Database-backed library websites: a case study of the use of PHP and MySQL at the University of Nottingham
Mike GardnerStephen Pinfield

Article information:

Users who downloaded this article also downloaded:

For Authors

If you would like to write for this, or any other Emerald publication, then please use our Emerald for Authors service information about how to choose which publication to write for and submission guidelines are available for all. Please visit www.emeraldinsight.com/authors for more information.

About Emerald www.emeraldinsight.com

Emerald is a global publisher linking research and practice to the benefit of society. The company manages a portfolio of more than 290 journals and over 2,350 books and book series volumes, as well as providing an extensive range of online products and additional customer resources and services.

Emerald is both COUNTER 4 and TRANSFER compliant. The organization is a partner of the Committee on Publication Ethics (COPE) and also works with Portico and the LOCKSS initiative for digital archive preservation.

Case study

Database-backed library websites: a case study of the use of PHP and MySQL at the University of Nottingham

MIKE GARDNER and STEPHEN PINFIELD

ABSTRACT

This paper discusses the advantages of database-backed websites and describes the model for a library website implemented at the University of Nottingham using open source software, PHP and MySQL. As websites continue to grow in size and complexity it becomes increasingly important to introduce automation to help manage them. It is suggested that a database-backed website offers many advantages over one built from static HTML pages. These include a consistency of style and content, the ability to present different views of the same data, devolved editing and enhanced security. The University of Nottingham Library Services website is described and issues surrounding its design, technological implementation and management are explored.

1. Introduction

As websites grow in complexity it is important to choose the right technology to deliver the information. Here we discuss the advantages of database-backed websites and our model for a library website at Nottingham University using open source software, PHP and MySQL. Dynamically-generated websites are easy to manage and offer significant advantages over those built with static HTML.

1.1 Redesign of the Library website

During the reorganisation of the University of Nottingham Library Services website (*http://www.nottingham.ac.uk/library*) in 1999, the decision was made to improve the design by minimising the number of options presented to the user on the home page. As well as options to find out more about the library, many library home pages offer a number of different choices to users, such as 'subject guides', 'networked CD-ROMs' and 'online databases'. These choices are often based on the format of the resources (CD-ROMs, websites, printed items). Monica Brinkley points out that this is "undoubtedly the least likely way the user will approach an information need, and should not be the primary route to the information they seek"[1].

In an attempt to reduce the potential for user confusion, it was decided to bring these resources together under one option: 'Subject resources' (the other home page options are currently 'Online catalogue', 'Services and information', 'Opening hours', 'Manuscripts and Special Collections' and 'Contact us'). The pages in the Subject resources section of the website offer a subject-based view of important resources, including bibliographies, databases and web gateways. Resources are listed together regardless of their format (although the format and access details of the resource are still made clear to the user for each resource). Staff and students know intuitively the subject in which they are interested and this section provides a simple gateway to the resources (as shown in Figure 1).

1.2 Static versus dynamic

It is fairly straightforward to produce web pages of lists of resources, their URLs and descriptions for different subject areas by using static HTML. The simplest way to produce such pages is to use a visual 'what-you-see-is-what-you-get' HTML editor, such as Microsoft FrontPage or Macromedia Dreamweaver. However, this is not an activity that scales well. Once there are several hundred resources, many of which may appear on a number of pages, inconsistencies begin to develop and maintenance becomes a time-consuming task.

The alternative is to use dynamic pages that are created on demand by

Figure 1. Subject resources homepage

34

pulling the requested information from a database. This offers several major advantages:

– a database, with a set of defined fields, helps to ensure consistency of content between records;
– one resource can appear on several different pages, however, since there is only one database entry behind these, consistency is ensured;
– unlike static HTML files, it is simple to search the database (for title keywords, for example);
– it is possible to present different views of the same data, producing subject pages as well as those based on format for users to browse;
– style is separated from content, it is therefore easy to ensure consistency and changing the style requires minimal effort (changes only need to be made once - to the template);
– editing can easily be distributed and controlled via web forms;
– security is easy to manage: password fields can be hidden to certain types of users (such as those viewing the pages from off-campus) if required.

For these reasons, it was decided to develop a web-enabled database to form the basis of the Subject resources section of the Library Services website. After some research into 'off the shelf' products, such as ROADS, it was also decided that they were overly complex for our needs and that a simple 'inhouse' solution would be preferable. ROADS (Resource Discovery and Indexing software) is a set of software tools that was developed as part of an Electronic Libraries Programme project to enable the set up and maintenance of web-based subject gateways (see *http://www.ilrt.bris.ac.uk/roads*). An inhouse system was seen as having the advantages of being simpler to install, maintain and use, without having to rely on external support. A locally developed system is also highly customisable: it can be designed to meet a particular need and further developed as needs change.

2.   Technologies

To deliver a database-backed website, three main technologies are required: a web server, a database and a mechanism for these two to interact and incorpo-
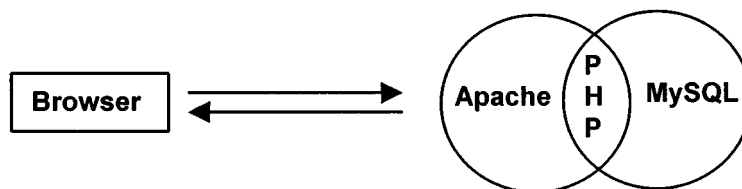

Figure 2. Interaction of Apache, MySQL and PHP

35

rate database content in the HTML pages. We chose to run the Apache web server (see *http://www.apache.org*), the most popular web server on the Internet[2] on a UNIX server. For a relational database, we chose MySQL (*http://www.mysql.com*) which is stable, has a very fast query response time on simple tables, is reliable and ANSI SQL92 compliant.[3] To glue these two together (see Figure 2), we chose the increasingly popular server-side HTML embedded scripting language PHP (see *http://www.php.net*). PHP was initially developed as a personal project, hence the original name Personal Home Page. It is now a major open source server-side HTML embedded scripting language and is in use at over two million websites, including Volvo (see *http://www.volvo.com*), Mitsubishi Motors (see *http://www.mitsucars.com*), W3C (see *http://www.w3.org*) and HumanitiesWeb (see *http://humanities web.org*). PHP is available as an Apache module, and is therefore very efficient, and works well with MySQL (as well as a variety of other databases).

PHP is very easy to integrate with HTML. Blocks of PHP code can simply be inserted in a standard HTML document as required. Before the web server sends the requested file to a user's browser it executes any PHP code within the page and incorporates the relevant output. For example, the PHP code <? print "Today's date is: date("M d, Y")"; ?> becomes Today's date is: August 1st, 2000. One of the most useful features of PHP is the way it handles HTML forms. Any form element automatically results in a variable with the same name being created on the target page. All the elements in the form are therefore available as variables and ready to incorporate in your HTML output or include in a database query. PHP can perform mathematical calculations, manipulate strings of data and (as in this case) query databases. It can work with many predefined functions, including XML support, e-mail capabilities (including IMAP functions) and automatic session handling.

All of these popular products are open source and available free of charge (see *http://www.opensource.org*). There is also support from the user community available in a variety of ways on the Internet. For example, there are excellent tutorials on PHP and MySQL at both the DevShed (see *http://www.devshed.com*) and Webreview (see *http://webreview.com*) sites, and active discussion boards on PHP at the PHPBuilder (see *http://www.phpbuilder.com*) and Faqts (see *http://www.faqts.com/knowledge-base/index.phtml/fid/51*) sites. The installation and setup of these products is not described here as this is done elsewhere.[4] However, it is possible to set up a standard Intel PC as a web/database UNIX server using Apache, PHP and MySQL in just a few hours. We chose to use Linux, a free version of UNIX. This combination of software running on a Linux server proved to be easy to install, tightly integrated, and stable. The whole system can also be assembled for very little cost. The Linux Documentation Project at Manchester Computing service provides a useful source of information about Linux (see *http://www.mcc.ac.uk/LDP*).

3. The 'Subject resources' database

3.1 Requirements

The Subject resources database contains metadata for various resources. The database was constructed in response to a number of specific requirements. The first requirement was that the various resources should be consistently described using a number of metadata elements. These were title, alternate title, author, URL, password/access details, and description (some of which would be optional) which equate to five key elements of the Dublin Core (see *http://purl.org/DC*). Secondly, it should be possible to associate each resource with one from a given list of 'sub-headings' (as listed in Appendix 1) and one or a number of 'format/category' details (as listed in Appendix 2). Thirdly, it should be possible to associate each entry with one or a number of different subjects, so that the same basic entry could be displayed under one or a number of subject pages. Fourthly, there was a requirement that *different* subject-specific comments could be associated with the *same* resource entry on *different* subject pages. For example, this would allow different comments to be made about Web of Science on the Chemistry page than under Economics. For each subject, it would also be required to assign a resource a 'status' of 'key resource' if appropriate.

Fifthly, there was also the requirement that the different entries in the database could be presented to the user in different ways. It should be possible to search for resources by title or title keyword. In addition, it should also be possible to browse a list of resources, divided under meaningful sub-headings, under a given subject. Subjects available largely correspond to the departments and schools in the University. It should also be possible (where appropriate) to find resources by format.

Finally, it was also required that maintenance of the database entries should be easily carried out by a number of subject librarians. Adding new resources and editing existing ones should be a straightforward task.

3.2 Data analysis

In a relational database multiple tables of data relate to each other through special key fields (this is in contrast to a flat file database which contains a single table of data). The main advantage of a relational, as opposed to a flat file, database is that data duplication, and therefore potential inconsistancies, are eliminated. In response to the requirements listed above, seven tables were designed within the Subject resources database to contain the metadata. These are outlined below and illustrated in Figure 3.

- resource - resource details (title, URL, description etc);
- subject - subjects and their introductions;
- subheading - subheadings that resources are listed under;
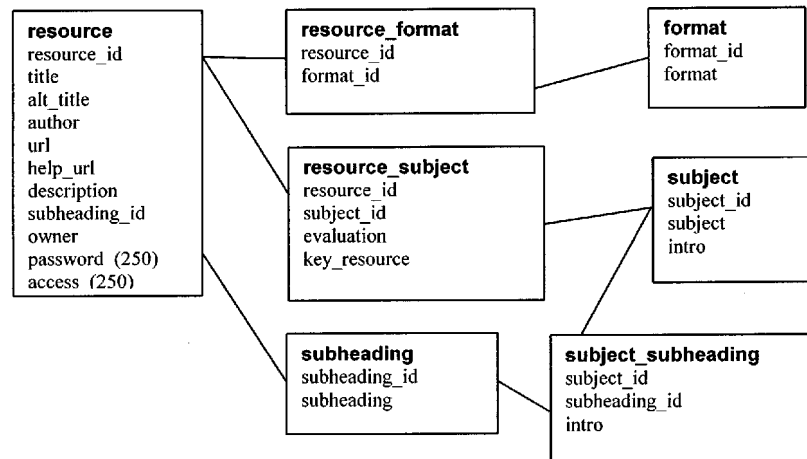- format - common formats and access methods;

37

Figure 3. Subject resources data model

- resource_subject - defines which resources appear on which subject pages;
- subject_subheading - subject specific sub-heading introductions;
- resource_format - defines which formats/access methods apply to each resource.

Some common formats of material (such as networked CD-ROMs) and access methods (such as 'ATHENS username and password required', or 'Access from the Nottingham campus only') require links to further information. (ATHENS is a UK authentication system used for accessing a number of databases see *http://www.athens.ac.uk*). Separating these options off, in a separate format table, enables many resources to include or link to one copy of this information which ensures consistency. Similarly the lists of sub-headings and subjects are located in separate tables.

3.3 The web pages

Users can browse the Subject resources by subject (from the menu page illustrated in Figure 1) or search for a particular resource by title. A typical subject page is shown in Figure 4 and comprises of:

- the title of the page;
- a general subject description;
- a menu of sub-headings together with a key to the icons used;
- key resources listed at the top of the page followed by all the resources associated with that subject, sorted alphabetically by sub-heading.

When the user selects a given subject from the Subject resources menu page, the subject page is constructed 'on the fly' from the database using PHP. The

38

Figure 4. Example subject page

results page is passed the required subject-id and a query of the subject table produces the full subject title and introduction for the page. A second query identifies all the required resource_ids related to this subject from the resource_subject table. All the details for each resource are then pulled from the resource table and printed out to the HTML page. The styles of the page itself are determined by the use of cascading style sheets (which are used throughout the Library Services website).

Each resource entry is displayed as illustrated in Figure 5 and consists of:

1. The resource title (clickable for appropriate electronic resources)
2. A set of icons indicating the resource format (such as print, CD-ROM, web) and access (such as ATHENS, campus access only) details
3. The URL (for web resources)
4. The description
5. Any special access requirements (passwords are only displayed to users accessing the page from on campus)
6. The subject specific evaluation (optional).

39

Program *vol. 35 no. 1*

Figure 5. Example resource entry

3.4   Administration

Subject librarians can all update the database via a series of password protected web forms (illustrated in Figure 6). This is simple to do and is a very efficient way of devolving the content creation to a large number of people whilst maintaining a consistent style.

4.   Electronic journals

The above model may appear very specific but it is actually extremely flexible and can easily be applied to other projects. We have used a similar approach to
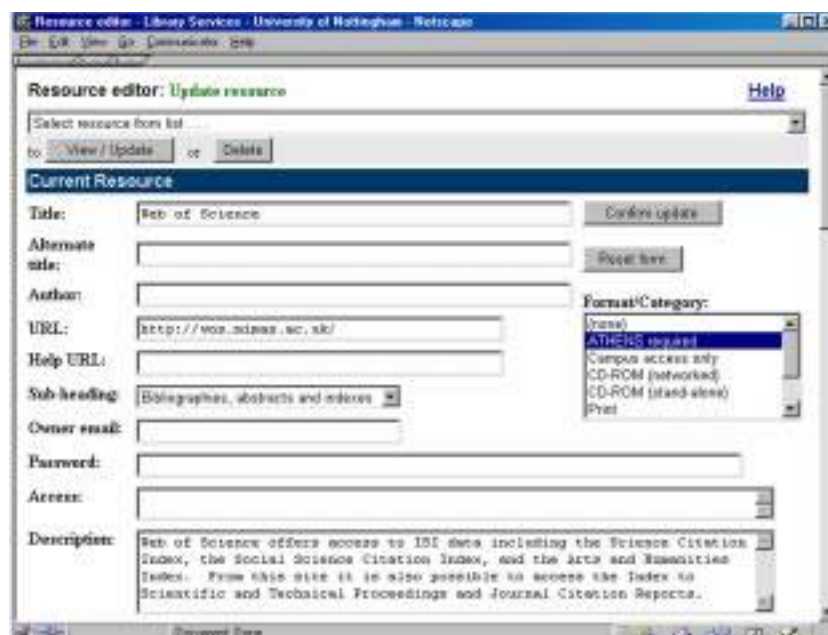

Figure 6. Updating a resource entry

40

generate our index of electronic journals (*http://www.nottingham.ac.uk/library/ejournals/*). This service offers subject-based or alphabetical lists of titles, and users can also search for particular titles. In addition, we have added a message field, to inform users about a change in the service, which can apply to an individual title or a whole package of titles from a particular supplier.

5. Conclusion

As library websites migrate from being about the service to being part of the service they inevitably grow in complexity. In this environment, some degree of automation is advantageous otherwise maintenance becomes difficult to manage. In many cases, automation can improve the content quality and navigability of the site. Preliminary evidence suggests that the Subject resources database is popular with both library staff and users. The former find it easy to maintain and useful in their information skills training and enquiry work. The latter are happy to work with a subject-based one-stop-shop approach to the quality information resources available to them.

References

1. Monica Brinkley, The future of library websites. *Vine*, no.113, 1999, pp.113-125.
2. The Netcraft web server survey September 2000 *http://www.netcraft.com/survey* (9th Oct. 2000).
3. Structured Query Language – an American National Standards Institute standard – more information from *http://www.jcc.com/SQLPages/jccs_sql.htm*
4. Linux setup information from *http://www.linux.com* and *http://www.mcc.ac.uk/LDP/* Apache, PHP, MySQL setup information from *http://www.devshed.com/*Server_Side/Administration/Database and from *http://www.vtwebwizard.com/tutorials/mysql/index.html*

Authors

Mike Gardner, Web Support Officer, Library Services, University of Nottingham, Nottingham, NG7 2RD, UK.
E-mail: mike.gardner@nottingham.ac.uk
Stephen Pinfield, Academic Services Librarian, Library Services, University of Nottingham, Nottingham, NG7 2RD, UK.
E-mail : stephen.pinfield@ nottingham.ac.uk

Appendix 1   'Subheadings' for the Subject resources pages

Bibliographies, abstracts and indexes
Careers
Catalogues of other libraries
Company information
Conferences
Courseware
Current awareness
Data
Full text resources
Law reports
Legislation
Mailing lists
Market research
Organisations
Other web resources
Patents, design, copyright
Reference sources
Statistics
Theses
Treaties
Web subject gateways

Appendix 2   'Format/category' for the Subject resources pages

ATHENS required
CD-ROM (networked)
CD-ROM (stand-alone)
Nottingham campus access only
Print
Sutton Bonington access only
Unrestricted access
Web resources

This article has been cited by:

1. Bibliography 319-344. [CrossRef]
2. Andrew Cox, Robin Yeates. 2003. Library portal solutions. *Aslib Proceedings* **55**:3, 155-165. [Abstract] [Full Text] [PDF]